HANOI UNIVERSITY OF SCIENCE
VIETNAM NATIONAL UNIVERSITY

Báo cáo
Xây dựng chatbot hỗ trợ học tập

Submitted by:
Lê Thái Dương - 21001541

Mục lục

Chương 1

Mở đầu

1.1 Tóm tắt dự án

Dự án xây dựng **Chatbot AI thông minh** hỗ trợ học sinh học Vật Lý 12, sử dụng công nghệ RAG (Retrieval-Augmented Generation) kết hợp với mô hình ngôn ngữ lớn Claude Sonnet 4.5. Hệ thống có khả năng trả lời câu hỏi dựa trên 4 chương sách giáo khoa (Vật lý nhiệt, Khí lý tưởng, Từ trường, và Vật lý hạt nhân), giải thích công thức và định nghĩa với ngôn ngữ dễ hiểu, đồng thời hiển thị công thức toán học bằng LaTeX một cách chuyên nghiệp. Chatbot còn có khả năng nhớ lịch sử hội thoại để trả lời các câu hỏi follow-up và tự động tìm kiếm thông tin liên quan trong tài liệu học tập.

Về mặt kỹ thuật, hệ thống đã xử lý thành công toàn bộ 4 chương SGK Vật Lý 12 và bộ tài liệu tự biên soạn thành 3,455 đoạn văn bản (chunks) và tích hợp hơn 70 công thức cơ bản vào prompt template. Thời gian phản hồi trung bình chỉ từ 2-3 giây, với chất lượng câu trả lời đạt 3.7/5 điểm trên 30 test cases đa dạng. Chi phí vận hành ước tính khoảng $0.02-0.05 cho mỗi câu hỏi khi sử dụng Claude API.

1.2 Bài toán đặt ra

1.2.1 Vấn đề thực tế

Qua quá trình quan sát và tìm hiểu, tôi nhận thấy học sinh lớp 12 đang gặp những khó khăn đáng kể trong việc học tập môn Vật Lý. Sách giáo khoa có hơn 200 trang với 4 chương lý thuyết phức tạp, khiến việc tìm kiếm thông tin trở nên tốn thời gian và kém hiệu quả. Mỗi lần cần tra cứu một công thức hay định nghĩa, các em phải lật tìm rất lâu vì không nhớ chính xác nội dung đó nằm ở trang nào, chương nào. Điều này đặc biệt gây khó khăn trong giờ làm bài tập và khi cần ôn thi nhanh.

Về công cụ hỗ trợ học tập, các giải pháp hiện có đều có những hạn chế riêng. Google Search tuy miễn phí nhưng cho kết quả chung chung, không dựa trên

chương trình SGK Việt Nam và thường lạc đề. Các ứng dụng học tập chỉ cung cấp bài giải mẫu cố định, không cho phép hỏi câu hỏi tự do. ChatGPT tuy thông minh nhưng không dựa vào tài liệu cụ thể nên hay "bịa đặt" thông tin (hiện tượng hallucination). Còn việc hỏi thầy cô thì bị giới hạn bởi thời gian, không thể hỏi được lúc nửa đêm hay cuối tuần khi đang tự học.

Áp lực ôn tập và thi cử càng làm vấn đề trở nên nghiêm trọng hơn. Gần kỳ thi, học sinh cần ôn lại nhiều chương cùng lúc nhưng dễ quên công thức và định nghĩa sau một thời gian không học. Do đó, các em rất cần có một "gia sư ảo" luôn sẵn sàng giải đáp 24/7, cho phép học theo tốc độ riêng mà không bị gò bó bởi lịch học cố định hay giờ giấc làm việc của giáo viên.

1.2.2 Ý nghĩa thực tiễn

Dự án này mang lại giá trị thiết thực cho nhiều đối tượng khác nhau trong hệ thống giáo dục. Đối với học sinh lớp 12, chatbot giúp tra cứu thông tin nhanh chóng chỉ trong 2-3 giây thay vì phải mất 5-10 phút lật sách, từ đó tập trung được vào việc hiểu khái niệm thay vì tìm kiếm. Các em có thể ôn tập linh hoạt bất cứ lúc nào, ở đâu chỉ cần có Internet, và luôn có một "trợ lý ảo" giải đáp thắc mắc ngay lập tức, giúp các em tự tin hơn trong học tập.

Đối với giáo viên, hệ thống giúp giảm tải công việc bằng cách tự động trả lời các câu hỏi lặp đi lặp lại. Thầy cô có thể xem thống kê các câu hỏi phổ biến của học sinh để điều chỉnh phương pháp giảng dạy cho phù hợp hơn. Chatbot hoạt động như một "trợ giảng ảo" hỗ trợ hiệu quả trong giờ tự học, giúp giáo viên có thêm thời gian tập trung vào các hoạt động sư phạm quan trọng khác.

Xét về phạm vi xã hội rộng lớn hơn, dự án có ý nghĩa quan trọng trong việc dân chủ hóa giáo dục. Nội dung chatbot được xây dựng hoàn toàn bằng tiếng Việt và phù hợp 100% với chương trình SGK Việt Nam, giúp học sinh ở vùng xa hay có hoàn cảnh khó khăn cũng có thể tiếp cận công nghệ AI hiện đại.

1.2.3 Đóng góp của dự án

Dự án này đóng góp vào lĩnh vực giáo dục và trí tuệ nhân tạo ở nhiều khía cạnh khác nhau. Về mặt kỹ thuật, dự án xây dựng thành công một hệ thống RAG hoàn chỉnh kết hợp ba công nghệ tiên tiến: Claude (mô hình ngôn ngữ), PhoBERT (embeddings tiếng Việt), và Pinecone (cơ sở dữ liệu vector trên cloud). Về mặt thực tiễn, dự án giải quyết được bài toán cụ thể của học sinh Việt Nam trong việc tra cứu và học tập môn Vật Lý 12.

Dự án cũng đóng góp cho cộng đồng bằng cách công khai toàn bộ mã nguồn trên GitHub, cho phép các nhà phát triển, sinh viên và học sinh khác có thể học hỏi và phát triển thêm. Đặc biệt, báo cáo được viết chi tiết bằng tiếng Việt với ngôn ngữ dễ hiểu, giúp người đọc có thể nắm bắt được cách thức hoạt động của

một hệ thống AI ứng dụng thực tế.

Về khả năng mở rộng, kiến trúc của dự án có thể dễ dàng áp dụng cho các môn học khác như Hóa học, Toán học, hay Sinh học, chỉ cần thay đổi nguồn dữ liệu đầu vào. Điều này mở ra tiềm năng phát triển một nền tảng học tập toàn diện hỗ trợ bởi AI cho toàn bộ chương trình THPT, góp phần nâng cao chất lượng giáo dục phổ thông tại Việt Nam.

Chương 2

Phương pháp và Triển khai

2.1 Phương pháp

2.1.1 Cách tiếp cận: RAG (Retrieval-Augmented Generation)

Để xây dựng chatbot có khả năng trả lời chính xác dựa trên sách giáo khoa, dự án này áp dụng phương pháp RAG (Retrieval-Augmented Generation). Phương pháp này kết hợp hai giai đoạn quan trọng: truy xuất thông tin từ tài liệu và sinh câu trả lời bằng mô hình ngôn ngữ lớn. Khác với các chatbot AI truyền thống chỉ dựa vào kiến thức được huấn luyện sẵn, RAG cho phép hệ thống "tham khảo" trực tiếp từ nguồn tài liệu đáng tin cậy trước khi đưa ra câu trả lời.

RAG giải quyết được hai vấn đề cốt lõi của các mô hình ngôn ngữ lớn hiện đại. Vấn đề đầu tiên là hiện tượng "ảo giác" (hallucination), khi mô hình như ChatGPT có thể tự tạo ra thông tin sai lệch vì không có nguồn tham chiếu cụ thể. Vấn đề thứ hai là giới hạn về độ dài ngữ cảnh, khiến mô hình không thể xử lý toàn bộ hàng trăm trang sách giáo khoa trong một lần. RAG khắc phục những hạn chế này bằng cách chỉ trích xuất những đoạn văn bản có độ liên quan cao nhất với câu hỏi, sau đó cung cấp chúng làm ngữ cảnh cho mô hình sinh câu trả lời.

Quy trình RAG trong dự án này được chia thành ba bước tuần tự:

1. **Embedding câu hỏi**: Câu hỏi của người dùng được chuyển đổi thành vector số học 768 chiều thông qua mô hình PhoBERT.
2. **Truy xuất tài liệu**: Hệ thống tìm kiếm trong cơ sở dữ liệu vector Pinecone để lấy ra 12 đoạn văn bản có độ tương đồng ngữ nghĩa cao nhất.
3. **Sinh câu trả lời**: Những đoạn văn bản được chọn kết hợp với câu hỏi và prompt template, rồi gửi đến Claude Sonnet 4.5 để tổng hợp thành câu trả lời hoàn chỉnh bằng tiếng Việt.

2.1.2 Cơ sở lý thuyết

Vector Embeddings và Cosine Similarity

Nền tảng của hệ thống truy xuất thông tin nằm ở khả năng biểu diễn văn bản dưới dạng vector trong không gian nhiều chiều. Trong dự án này, mỗi đoạn văn bản từ sách giáo khoa hoặc câu hỏi của người dùng đều được chuyển đổi thành một vector số thực 768 chiều thông qua mô hình PhoBERT. Vector embedding này không chỉ mã hóa chuỗi ký tự mà còn nắm bắt được ý nghĩa ngữ nghĩa sâu sắc của văn bản.

Để đo lường độ tương đồng giữa hai đoạn văn bản, hệ thống sử dụng độ đo cosine similarity, được định nghĩa như sau:

$$\text{similarity}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{|\mathbf{u}| \cdot |\mathbf{v}|} = \frac{\sum_{i=1}^{768} u_i v_i}{\sqrt{\sum_{i=1}^{768} u_i^2} \cdot \sqrt{\sum_{i=1}^{768} v_i^2}}$$

Giá trị similarity nằm trong khoảng $[-1, 1]$, trong đó giá trị 1 biểu thị hai vector có hướng hoàn toàn giống nhau (nghĩa tương đồng cao), giá trị 0 cho biết hai vector trực giao (không liên quan), và giá trị -1 cho thấy hai vector ngược hướng hoàn toàn. Trong thực tế, các đoạn văn bản liên quan thường có cosine similarity trong khoảng 0.7 đến 0.95.

Maximum Marginal Relevance (MMR)

Khi truy xuất tài liệu chỉ dựa trên độ tương đồng cao nhất, hệ thống dễ rơi vào tình trạng lấy nhiều đoạn văn có nội dung trùng lặp hoặc quá tương tự nhau. Điều này làm giảm độ phong phú của thông tin cung cấp cho mô hình ngôn ngữ. Thuật toán Maximum Marginal Relevance (MMR) ra đời để giải quyết vấn đề này bằng cách tối ưu hóa đồng thời hai mục tiêu: đảm bảo độ liên quan cao với câu hỏi và tăng tính đa dạng giữa các đoạn văn bản được chọn.

Công thức toán học của MMR được biểu diễn như sau:

$$\text{MMR} = \arg\max_{D_i \in R \setminus S} \left[\lambda \cdot \text{Sim}_1(D_i, Q) - (1 - \lambda) \cdot \max_{D_j \in S} \text{Sim}_2(D_i, D_j) \right]$$

Trong công thức trên, Q đại diện cho câu hỏi của người dùng, R là tập 30 đoạn văn bản ứng viên có độ tương đồng ban đầu cao nhất được lấy từ cơ sở dữ liệu vector, và S là tập các đoạn văn bản đã được chọn ở các bước trước đó. Hệ số λ (trong dự án này được đặt bằng 0.5) quyết định mức độ cân bằng giữa hai yếu tố: $\text{Sim}_1(D_i, Q)$ đo độ liên quan của đoạn văn bản ứng viên với câu hỏi, trong khi $\max_{D_j \in S} \text{Sim}_2(D_i, D_j)$ đo độ trùng lặp cao nhất với các đoạn đã chọn.

Thuật toán hoạt động theo cơ chế lặp: ở mỗi bước, nó tính điểm MMR cho tất cả các đoạn văn bản chưa được chọn, sau đó chọn đoạn có điểm số cao nhất. Quá trình này tiếp tục cho đến khi đủ 12 đoạn văn bản được lựa chọn. Với $\lambda = 0.5$, hệ thống cân bằng đều giữa việc lấy thông tin liên quan và đảm bảo tính đa dạng, giúp mô hình ngôn ngữ có được cái nhìn toàn diện hơn về vấn đề được hỏi.

Hybrid Search: Kết hợp Vector Search và BM25

Để nâng cao độ chính xác trong quá trình truy xuất thông tin, dự án triển khai phương pháp hybrid search, kết hợp điểm mạnh của hai cách tiếp cận khác nhau. Phương pháp đầu tiên là vector search (semantic search), hoạt động dựa trên việc so sánh ý nghĩa ngữ nghĩa giữa các đoạn văn bản. Cách tiếp cận này đặc biệt hiệu quả khi câu hỏi và tài liệu sử dụng từ ngữ khác nhau nhưng cùng ý nghĩa, ví dụ như "định luật" và "nguyên lý", hoặc "nhiệt độ" và "độ nóng".

Phương pháp thứ hai là BM25 (Best Matching 25), một thuật toán tìm kiếm từ khóa được cải tiến từ TF-IDF. BM25 tính điểm liên quan dựa trên tần suất xuất hiện của các từ khóa trong tài liệu, có tính đến độ dài văn bản và tính phổ biến của từ trong toàn bộ tập tài liệu. Công thức toán học của BM25 được biểu diễn:

$$\text{BM25}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i,D) \cdot (k_1+1)}{f(q_i,D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{\text{avgdl}})}$$

Trong công thức này, $f(q_i,D)$ biểu thị tần suất xuất hiện của từ q_i trong tài liệu D, còn $|D|$ là số lượng từ trong tài liệu đó. Giá trị avgdl đại diện cho độ dài trung bình của tất cả tài liệu trong cơ sở dữ liệu. Hai tham số điều chỉnh k_1 và b thường được đặt lần lượt là 1.5 và 0.75, trong đó k_1 kiểm soát mức độ bão hòa của tần suất từ và b điều chỉnh mức độ ảnh hưởng của độ dài tài liệu.

Điểm số cuối cùng của hybrid search được tính bằng cách kết hợp tuyến tính hai điểm từ hai phương pháp:

$$\text{score}_{\text{hybrid}} = 0.7 \cdot \text{score}_{\text{vector}} + 0.3 \cdot \text{score}_{\text{BM25}}$$

Tỷ lệ 70-30 này được chọn dựa trên thực nghiệm, ưu tiên vector search vì khả năng hiểu ngữ nghĩa tốt hơn, đồng thời vẫn giữ lại ưu điểm của tìm kiếm từ khóa chính xác từ BM25. Cách tiếp cận hybrid này đặc biệt hiệu quả với các câu hỏi chứa thuật ngữ chuyên môn hoặc công thức cụ thể.

2.1.3 Dữ liệu sử dụng

Nguồn dữ liệu

Dữ liệu đào tạo cho chatbot được lấy từ bộ tài liệu "Phong tỏa vật lí 12", một nguồn tài liệu tham khảo toàn diện được biên soạn dựa trên ba bộ sách giáo khoa Vật Lý 12 chính thức của Bộ Giáo dục và Đào tạo Việt Nam. Ba bộ sách này bao gồm: bộ Kết nối tri thức với cuộc sống, bộ Chân trời sáng tạo, và bộ Cánh diều. Việc tổng hợp từ nhiều nguồn giúp đảm bảo tính đầy đủ và đa dạng của nội dung, đồng thời tránh thiên lệch về phong cách trình bày của một bộ sách cụ thể.

Bộ tài liệu "Phong tỏa vật lí 12" được cấu trúc thành bốn phần chính tương ứng với bốn chương trọng tâm trong chương trình Vật Lý lớp 12:

- **Chương 1: Vật lý nhiệt** – Nhiệt động lực học, khái niệm nội năng, các quá trình nhiệt động (đẳng nhiệt, đẳng áp, đẳng tích).
- **Chương 2: Khí lý tưởng** – Phương trình trạng thái, định luật Gay-Lussac và Boyle-Mariotte.
- **Chương 3: Từ trường** – Lực Lorentz, cảm ứng từ, từ thông, định luật Faraday về cảm ứng điện từ.
- **Chương 4: Vật lý hạt nhân** – Các hiện tượng phóng xạ, phản ứng hạt nhân, năng lượng liên kết.

Toàn bộ tài liệu được số hóa thành bảy file PDF, với tổng dung lượng vượt quá 200 trang nội dung chi tiết. Bốn file đầu tương ứng với bốn chương chính của "Phong tỏa vật lí 12", trong khi ba file còn lại chứa nội dung từ ba bộ sách giáo khoa gốc. Sự kết hợp này tạo ra một cơ sở kiến thức phong phú, đảm bảo chatbot có thể trả lời được đa dạng các câu hỏi từ nhiều góc độ tiếp cận khác nhau.

Quy trình xử lý dữ liệu

Quy trình xử lý dữ liệu được thiết kế theo pipeline bốn bước, mỗi bước đều quan trọng trong việc chuyển đổi tài liệu PDF thô thành cơ sở dữ liệu vector có thể tìm kiếm hiệu quả:

Bước 1: Trích xuất văn bản từ PDF. Thư viện `pypdf` được sử dụng để đọc từng trang PDF và thu thập toàn bộ nội dung văn bản. Thư viện này được lựa chọn vì khả năng xử lý tốt các ký tự tiếng Việt có dấu cũng như các ký tự đặc biệt thường xuất hiện trong công thức toán học và vật lý. Quá trình trích xuất diễn ra tuần tự qua từng trang, đảm bảo thứ tự nội dung được giữ nguyên so với tài liệu gốc.

Bước 2: Phân đoạn văn bản (Chunking). Văn bản dài từ mỗi file PDF được chia nhỏ bằng thuật toán `RecursiveCharacterTextSplitter` với kích thước

chunk là 1000 ký tự. Điểm đặc biệt của quá trình này là kỹ thuật overlap, trong đó mỗi chunk sẽ chồng lấn 200 ký tự với chunk kế tiếp. Kỹ thuật overlap này đóng vai trò then chốt trong việc đảm bảo các khái niệm, định nghĩa hay công thức nằm ở ranh giới giữa hai chunks không bị cắt đứt và mất ngữ cảnh. Điều này giúp mô hình có thể hiểu đầy đủ ý nghĩa ngay cả khi thông tin cần thiết trải dài qua nhiều đoạn văn.

Bước 3: Trích xuất và phân tích hình ảnh (tùy chọn). Đối với những trang chứa nhiều hình vẽ, biểu đồ, hoặc đồ thị, hệ thống sử dụng thư viện `pdf2image` (yêu cầu công cụ Poppler) để chuyển đổi trang PDF thành hình ảnh PNG. Sau đó, hình ảnh được gửi đến Claude Vision API để nhận được mô tả chi tiết bằng ngôn ngữ tự nhiên về nội dung hình ảnh. Mô tả này bao gồm thông tin về biểu đồ, công thức viết tay, sơ đồ mạch điện hay các hình minh họa khác. Cuối cùng, mô tả văn bản được nhúng vào chunk tương ứng, giúp chatbot có thể trả lời các câu hỏi liên quan đến hình ảnh như "Giải thích đồ thị P-V trong quá trình đẳng nhiệt" hay "Mô tả sơ đồ thí nghiệm về cảm ứng điện từ".

Bước 4: Embedding và lưu trữ. Mỗi chunk văn bản được đưa qua mô hình PhoBERT để tạo ra một vector 768 chiều, sau đó được lưu trữ trong Pinecone cùng với metadata bao gồm tên file PDF gốc, số thứ tự chunk, và nội dung văn bản đầy đủ. Sau khi hoàn thành quy trình xử lý cho toàn bộ bảy file PDF, hệ thống tạo ra tổng cộng 3,455 vectors được đánh index trong Pinecone, sẵn sàng phục vụ cho các truy vấn tìm kiếm với độ trễ thấp.

2.1.4 Mô hình và công nghệ

PhoBERT - Mô hình Embedding cho tiếng Việt

PhoBERT, với định danh đầy đủ là VoVanPhuc/sup-SimCSE-VietNamese-phobert-base, đóng vai trò cốt lõi trong việc chuyển đổi văn bản tiếng Việt thành các biểu diễn vector số học. Đây là một biến thể của mô hình BERT được huấn luyện trước trên một corpus tiếng Việt lớn, sau đó được tinh chỉnh đặc biệt cho tác vụ đo lường độ tương đồng câu. Các đặc điểm kỹ thuật chính:

- **Kích thước vector**: 768 chiều
- **Tokenizer**: Word segmentation cho tiếng Việt
- **Hiệu suất**: Tốt hơn 15-20% so với mô hình multilingual trên tác vụ semantic search tiếng Việt

Điểm mạnh đặc biệt của PhoBERT nằm ở tokenizer được thiết kế riêng cho tiếng Việt. Khác với các mô hình đa ngôn ngữ sử dụng tokenization dựa trên ký tự hoặc subword chung cho nhiều ngôn ngữ, PhoBERT áp dụng word segmentation - quá trình tách từ phù hợp với đặc điểm của tiếng Việt. Điều này giúp mô hình

nắm bắt chính xác hơn ranh giới từ và ý nghĩa ngữ cảnh, đặc biệt quan trọng trong một ngôn ngữ có nhiều từ ghép và đơn vị nghĩa phức tạp như tiếng Việt.

Trong các thử nghiệm so sánh, PhoBERT cho thấy hiệu suất vượt trội so với các mô hình embedding đa ngôn ngữ phổ biến. So với mô hình `MiniLM-L12-v2` có kích thước vector 384 chiều, PhoBERT đạt kết quả tốt hơn 15-20% trên các tác vụ tìm kiếm ngữ nghĩa tiếng Việt. Sự cải thiện này đến từ cả việc được huấn luyện chuyên biệt trên dữ liệu tiếng Việt và có không gian embedding lớn hơn (768 so với 384 chiều), cho phép biểu diễn chi tiết hơn các nét nghĩa tinh tế của ngôn ngữ.

Claude Sonnet 4.5 - Mô hình Ngôn ngữ Lớn

Claude Sonnet 4.5, với mã định danh `claude-sonnet-4-5`, là mô hình ngôn ngữ lớn tiên tiến được phát triển bởi Anthropic. Mô hình này có context window lên đến 200,000 tokens, cho phép xử lý và hiểu được một lượng lớn thông tin ngữ cảnh trong một lần. Khả năng này đặc biệt quan trọng trong ứng dụng RAG, nơi mô hình cần đọc và tổng hợp thông tin từ 12 đoạn văn bản dài, mỗi đoạn có thể chứa tới hàng trăm từ. Đầu ra của mô hình có thể lên đến 8,192 tokens, đủ để tạo ra các câu trả lời chi tiết và toàn diện.

Các thông số kỹ thuật chính:

- **Context window**: 200,000 tokens
- **Output tối đa**: 8,192 tokens
- **Temperature**: 0.3 (ưu tiên độ chính xác)
- **Điểm mạnh**: Reasoning tốt, ít hallucination, hỗ trợ tiếng Việt tự nhiên

Điểm nổi bật của Claude Sonnet 4.5 nằm ở khả năng reasoning (suy luận) mạnh mẽ và tỷ lệ hallucination thấp. Trong khi nhiều mô hình ngôn ngữ lớn khác có xu hướng tự tạo ra thông tin không có thật khi thiếu kiến thức, Claude được thiết kế để nhận biết giới hạn của mình và dựa chặt chẽ vào ngữ cảnh được cung cấp. Khả năng hỗ trợ tiếng Việt tự nhiên của mô hình cũng rất ấn tượng, tạo ra các câu trả lời với ngữ pháp chính xác và phong cách phù hợp với văn hóa Việt Nam. Trong cấu hình của dự án, temperature được đặt ở mức 0.3, một giá trị thấp giúp mô hình tập trung vào độ chính xác và tính nhất quán thay vì sự sáng tạo.

Trong hệ thống chatbot này, Claude đóng vai trò như một "trợ giảng" với các khả năng:

- Hiểu và phân tích ngữ cảnh từ nhiều đoạn văn bản khác nhau
- Tổng hợp thông tin thành câu trả lời mạch lạc và dễ hiểu
- Render các công thức LaTeX một cách chính xác
- Phát hiện và từ chối câu hỏi ngoài phạm vi Vật Lý 12

Pinecone - Cơ sở Dữ liệu Vector

Pinecone được lựa chọn làm cơ sở dữ liệu vector cho dự án nhờ kiến trúc serverless được quản lý hoàn toàn trên đám mây. Cấu hình kỹ thuật:

- **Kiến trúc**: Serverless, tự động scale
- **Region**: AWS us-east-1
- **Metric**: Cosine similarity
- **Latency**: < 100ms mỗi query
- **Index name**: `studychatbot`

Khác với các giải pháp cơ sở dữ liệu vector truyền thống đòi hỏi việc thiết lập và bảo trì hạ tầng phức tạp, Pinecone tự động mở rộng quy mô dựa trên nhu cầu sử dụng mà không cần can thiệp thủ công.

Hệ thống sử dụng cosine similarity làm độ đo khoảng cách giữa các vectors, một lựa chọn phù hợp vì nó đo lường góc giữa các vectors thay vì khoảng cách Euclidean tuyệt đối. Điều này đặc biệt hiệu quả với các embeddings từ mô hình ngôn ngữ, nơi hướng của vector quan trọng hơn độ lớn của nó. Với kiến trúc được tối ưu hóa, Pinecone đạt được độ trễ dưới 100 miligiây cho mỗi truy vấn, đủ nhanh để tạo ra trải nghiệm tương tác mượt mà cho người dùng. Index được đặt tên là `studychatbot`, chứa toàn bộ 3,455 vectors tương ứng với các chunks từ tài liệu "Phong tỏa vật lí 12".

Quyết định chọn Pinecone thay vì các giải pháp local như FAISS được đưa ra dựa trên nhiều yếu tố kỹ thuật và thực tiễn. Pinecone loại bỏ nhu cầu lưu trữ file index có kích thước lớn trên máy local, điều đặc biệt quan trọng khi số lượng vectors tăng lên hàng triệu. Hệ thống hỗ trợ metadata filtering hiệu quả, cho phép lọc kết quả tìm kiếm dựa trên các thuộc tính như tên file nguồn hay số chương, giúp tinh chỉnh độ chính xác của retrieval. Khả năng mở rộng dễ dàng cũng là một ưu điểm lớn - khi cần thêm dữ liệu từ các môn học khác như Hóa học hay Toán học, chỉ cần upload thêm vectors mà không lo về hiệu suất. Cuối cùng, API của Pinecone đơn giản và được tích hợp sẵn trong framework LangChain, giảm đáng kể thời gian phát triển và bảo trì hệ thống.

2.2 Triển khai

2.2.1 Kiến trúc hệ thống

Hệ thống được xây dựng dựa trên kiến trúc ba tầng (3-tier architecture), một mô hình thiết kế phổ biến trong phát triển ứng dụng web hiện đại. Tầng trình diễn

(Presentation Layer) bao gồm giao diện web được xây dựng bằng HTML, CSS và JavaScript, tích hợp thư viện KaTeX để render các công thức toán học. Tầng ứng dụng (Application Layer) sử dụng Flask server để xử lý các API requests và điều phối toàn bộ RAG pipeline. Tầng dữ liệu (Data Layer) kết nối với Pinecone vector database để truy xuất thông tin và Claude API để sinh câu trả lời.

Luồng xử lý một câu hỏi từ người dùng diễn ra qua các bước sau:

1. Frontend gửi HTTP POST request đến `/api/ask`
2. Flask server nhận câu hỏi + lịch sử hội thoại
3. PhoBERT chuyển câu hỏi thành vector 768 chiều
4. Pinecone tìm kiếm 30 chunks có độ tương đồng cao nhất
5. Thuật toán MMR chọn 12 chunks đa dạng nhất
6. Prompt Builder kết hợp: 12 chunks + câu hỏi + 70 công thức cơ bản
7. Claude API sinh câu trả lời bằng tiếng Việt (có LaTeX)
8. Flask đóng gói JSON: {answer, sources}
9. Frontend render Markdown + KaTeX
10. Hiển thị cho người dùng (tổng thời gian: 2-3 giây)

2.2.2 Công cụ và thư viện

Backend Stack

Phần backend của hệ thống được xây dựng trên nền tảng Flask 3.1.1, một web framework Python nhẹ và đơn giản, đặc biệt phù hợp cho việc phát triển REST API. Thành phần trung tâm là LangChain 0.3.26, một framework orchestration chuyên dụng cho các ứng dụng RAG. LangChain cung cấp ConversationalRetrievalChain để xử lý chat có khả năng nhớ ngữ cảnh, `ConversationBufferMemory` để lưu trữ lịch sử hội thoại với giới hạn 2000 tokens, và `PineconeVectorStore` làm wrapper cho Pinecone client. Thư viện `langchain-anthropic` đóng vai trò cầu nối giữa Claude và LangChain.

Xử lý embeddings được thực hiện bởi thư viện `sentence-transformers`, cho phép load và sử dụng mô hình PhoBERT một cách hiệu quả. Quá trình đọc và trích xuất văn bản từ PDF sử dụng `pypdf`, trong khi cặp thư viện `pdf2image` và `Pillow` chịu trách nhiệm chuyển đổi các trang PDF thành hình ảnh để gửi đến Claude Vision API. Để hỗ trợ hybrid search, hệ thống tích hợp `rank-bm25` cho thuật toán BM25 và `jieba` để thực hiện word segmentation cho tiếng Việt.

Frontend Stack

Giao diện người dùng được xây dựng với phương pháp tối giản, sử dụng Vanilla JavaScript thay vì các framework phức tạp như React hay Vue, giúp giữ cho frontend nhẹ nhàng và nhanh chóng. Thư viện `Marked.js` đảm nhận việc parse Markdown thành HTML, hỗ trợ các định dạng như danh sách, in đậm, in nghiêng và code blocks. Đặc biệt quan trọng là `KaTeX`, một thư viện render công thức LaTeX thành SVG/HTML với chất lượng cao và tốc độ nhanh. CSS3 được sử dụng để tạo ra giao diện đẹp mắt với gradient background, animations mượt mà và thiết kế responsive thích ứng với nhiều kích thước màn hình.

Development Tools

Trong quá trình phát triển, Jupyter Notebook đóng vai trò quan trọng cho việc thử nghiệm và debug các thuật toán retrieval trước khi tích hợp vào hệ thống chính. Thư viện `python-dotenv` được sử dụng để quản lý API keys một cách an toàn, tránh việc hardcode thông tin nhạy cảm vào source code. Git được sử dụng làm hệ thống version control, với repository được lưu trữ trên GitHub để theo dõi lịch sử thay đổi và hỗ trợ cộng tác.

2.2.3 Cấu trúc mã nguồn

Dự án được tổ chức theo cấu trúc module rõ ràng với sự phân tách trách nhiệm hợp lý, giúp dễ dàng bảo trì và mở rộng. Thư mục `src/` chứa toàn bộ source code cốt lõi, bao gồm ba module Python chính.

Module `helper.py` là trung tâm của hệ thống. Module này chứa các hàm tiện ích như `load_documents()` để đọc và phân đoạn PDF, `process_images()` để trích xuất hình ảnh với Claude Vision, `create_chatbot()` để xây dựng RAG chain, và `get_response()` để xử lý câu hỏi.

Module `prompt.py` định nghĩa các prompt templates, trong đó có system prompt chứa hơn 70 công thức Vật Lý cơ bản và các hướng dẫn cho mô hình.

Module `advanced_retrieval.py` triển khai các thuật toán retrieval nâng cao như MMR, re-ranking với LLM, hybrid search kết hợp vector và BM25, cùng với query expansion.

File `app.py` là điểm vào của ứng dụng web, Flask server với ba endpoints chính: GET `/` để phục vụ giao diện HTML, POST `/api/ask` để xử lý câu hỏi, và POST `/api/rebuild` để rebuild index khi cần.

Script `upload_to_pinecone.py` thực hiện toàn bộ data pipeline: load PDFs từ thư mục data, phân đoạn documents, tạo PhoBERT embeddings và upload lên Pinecone với metadata đầy đủ.

Module đánh giá `evaluate.py` chứa 30 tescases phân loại theo nhiều tiêu chí,

hỗ trợ cả keyword-based evaluation miễn phí và LLM-based evaluation chi tiết hơn.

Thư mục `templates/` chứa file `index.html` này xây dựng giao diện chat với bubbles, tích hợp Marked.js để parse Markdown, KaTeX để tự động render công thức, và AJAX calls để giao tiếp với backend.

Thư mục `research/` chứa Jupyter notebook để thử nghiệm các thuật toán. Thư mục `data/` (được gitignore) lưu trữ bảy file PDF từ bộ "Phong tỏa vật lí 12" và ba bộ sách giáo khoa.

File `.env` (cũng được gitignore) chứa các API keys nhạy cảm cho Anthropic và Pinecone. Các file còn lại như `requirements.txt`, `README.md`, `BAO_CAO.md` và `DATA_SETUP.md` cung cấp documentation và hướng dẫn setup.

2.2.4 Module chính và hoạt động

Module helper.py

File `helper.py` đóng vai trò trung tâm trong hệ thống với các hàm xử lý cốt lõi.

Hàm `load_documents()` nhận vào đường dẫn thư mục chứa PDFs và một flag boolean để bật/tắt tính năng Claude Vision. Hàm này lặp qua tất cả file PDF trong thư mục, sử dụng `pypdf.PdfReader` để extract text. Nếu chế độ vision được bật, hàm còn extract và phân tích hình ảnh trước khi split text thành chunks bằng `RecursiveCharacterTextSplitter`. Mỗi chunk được gắn metadata bao gồm tên file nguồn và số thứ tự, sau đó được đóng gói thành danh sách Document objects.

Hàm `create_chatbot()` khởi tạo `ConversationalRetrievalChain`, thành phần điều phối toàn bộ quá trình RAG. Hàm nhận vào một retriever từ Pinecone và một conversation memory buffer.

Chain được cấu hình với Claude Sonnet 4.5 làm LLM, temperature 0.3 để ưu tiên độ chính xác, giới hạn đầu ra 2000 tokens. Chain sử dụng strategy "stuff" (nối tất cả chunks lại thành một prompt duy nhất). Cấu hình quan trọng khác là bật chế độ trả về source documents, cho phép người dùng xác minh nguồn gốc thông tin.

Hàm `get_response()` xử lý luồng tương tác thực tế với người dùng. Hàm nhận câu hỏi và session ID để track cuộc hội thoại, sau đó retrieve lịch sử chat từ memory.

Chain được invoke với câu hỏi hiện tại cộng với toàn bộ chat history, cho phép hiểu được ngữ cảnh các câu hỏi follow-up. Sau khi nhận phản hồi từ Claude, hàm parse câu trả lời và danh sách sources, cập nhật memory với cặp Q&A mới, và trả về một dictionary chứa answer cùng source documents.

Module advanced_retrieval.py

Module `advanced_retrieval.py` triển khai các kỹ thuật retrieval nâng cao.

Hàm `mmr_retrieval()` nhận vào vectorstore, câu hỏi và số lượng chunks cần lấy (mặc định 12). Thuật toán bắt đầu bằng việc fetch 30 candidates có độ tương đồng cao nhất từ vectorstore. Sau đó, một vòng lặp chạy k lần để lần lượt chọn từng chunk.

Ở mỗi vòng lặp, với mỗi candidate chưa được chọn, thuật toán tính điểm relevant bằng similarity với câu hỏi, tính điểm redundant bằng similarity cao nhất với các chunks đã chọn, rồi tổng hợp thành MMR score theo công thức $\lambda \cdot \text{relevant} - (1 - \lambda) \cdot \text{redundant}$. Candidate có MMR score cao nhất được thêm vào tập selected. Quá trình lặp lại cho đến khi đủ k chunks.

Hàm `hybrid_search()` kết hợp vector search và BM25 keyword search. Hàm nhận vectorstore, một BM25 index đã được xây dựng trước, câu hỏi và số lượng chunks cần trả về.

Đầu tiên, vector search được thực hiện để lấy 20 chunks với điểm similarity cao nhất. Đồng thời, BM25 search cũng tính điểm cho tất cả documents dựa trên tần suất từ khóa. Hai tập điểm số này được chuẩn hóa về khoảng [0, 1] để đảm bảo công bằng.

Điểm cuối cùng của mỗi document là tổ hợp tuyến tính với trọng số 0.7 cho vector score và 0.3 cho BM25 score. Các documents được sắp xếp theo điểm tổng hợp và k documents đứng đầu được trả về. Cách tiếp cận hybrid này tận dụng được cả khả năng hiểu ngữ nghĩa của vector search và độ chính xác của keyword matching.

API Flask

Endpoint chính của ứng dụng là POST `/api/ask`, xử lý toàn bộ logic tương tác với người dùng. Request gửi đến endpoint này chứa một JSON object với trường `question` bắt buộc và trường `session_id` tùy chọn để tracking cuộc hội thoại.

Khi nhận request, server parse JSON và kiểm tra tính hợp lệ của câu hỏi. Session ID được lấy từ request hoặc tạo mới nếu chưa có, sau đó conversation memory tương ứng được load. Hàm `get_response()` được gọi với chain, câu hỏi và session ID, trả về câu trả lời cùng danh sách source documents. Server extract thông tin sources và đóng gói thành JSON response với cấu trúc chứa `answer` và `sources`.

Hệ thống xử lý lỗi được thiết kế cẩn thận với nhiều tầng bảo vệ. Nếu request thiếu trường question, server trả về HTTP 400 Bad Request. Các lỗi từ Anthropic API như invalid key hoặc quota exceeded được catch và trả về HTTP 500 với thông báo lỗi chi tiết.

Đặc biệt, khi Pinecone timeout xảy ra do mạng chậm, hệ thống tự động retry

tối đa 3 lần với exponential backoff trước khi báo lỗi cho người dùng. Cơ chế này đảm bảo tính ổn định và trải nghiệm mượt mà cho người dùng cuối.

Giao diện web

Giao diện web được thiết kế với cấu trúc HTML đơn giản nhưng hiệu quả. Phần tử container chính chứa ba thành phần: header hiển thị tiêu đề chatbot, div `chat-container` làm khu vực hiển thị các bong bóng chat, và input-container chứa ô nhập liệu cùng nút gửi. Các thư viện Marked.js và KaTeX được load thông qua CDN, sau đó JavaScript được nhúng trực tiếp vào trang để xử lý logic tương tác.

Logic JavaScript được xây dựng xung quanh hai hàm chính. Hàm `sendMessage()` là async function được kích hoạt khi người dùng nhấn nút gửi hoặc Enter. Hàm này lấy câu hỏi từ input, hiển thị ngay lập tức dưới dạng bubble của người dùng, sau đó gửi POST request đến `/api/ask` với câu hỏi được JSON-encoded. Response được await và parse thành JSON object chứa answer và sources.

Hàm `displayBotMessage()` xử lý việc hiển thị câu trả lời từ bot. Đầu tiên, Markdown trong answer được parse thành HTML thông qua `marked.parse()`. Bubble được tạo ra và chèn HTML vào. Sau đó, hàm `renderMathInElement()` từ KaTeX được gọi để tự động phát hiện và render tất cả công thức toán học trong bubble. Hàm này nhận cấu hình về delimiters, trong đó $$ được dùng cho display math (công thức độc lập trên một dòng) và $ cho inline math (công thức trong đoạn văn). Cuối cùng, danh sách sources được append vào cuối bubble và toàn bộ bubble được thêm vào chat container, tạo ra hiệu ứng chat mượt mà.

2.2.5 Quy trình triển khai

Quá trình triển khai hệ thống được chia thành ba giai đoạn chính: setup môi trường, upload dữ liệu và chạy ứng dụng. Giai đoạn đầu tiên bắt đầu với việc clone repository từ GitHub về máy local và di chuyển vào thư mục dự án. Tiếp theo, toàn bộ Python dependencies được cài đặt thông qua file `requirements.txt`. Một bước quan trọng là cài đặt Poppler, công cụ cần thiết để chuyển đổi PDF sang image cho Claude Vision. Trên Windows, Poppler cần download thủ công từ repository oschwartz10612, trong khi trên Mac và Linux có thể cài qua package manager. Cuối cùng, file `.env` được tạo từ template và điền các API keys cho Anthropic và Pinecone.

Giai đoạn upload dữ liệu bắt đầu với việc copy các file PDF từ bộ "Phong tỏa vật lí 12" vào thư mục `data/`. Script `upload_to_pinecone.py` sau đó được chạy để xử lý toàn bộ pipeline. Script này load tất cả PDFs, hiển thị progress bar trong quá trình chunking, tạo embeddings cho 3,455 chunks và upload chúng lên Pinecone index tên `studychatbot`. Quá trình này thường mất khoảng 2-3 phút tùy vào tốc độ mạng và số lượng tài liệu.

Giai đoạn cuối cùng là chạy ứng dụng. Trong môi trường development, lệnh `python app.py` đơn giản sẽ khởi động Flask server trên địa chỉ local port 5000. Đối với môi trường production, hệ thống sử dụng gunicorn - một WSGI HTTP server production-ready. Gunicorn được cấu hình chạy 4 worker processes để xử lý đồng thời nhiều requests và bind vào địa chỉ public để có thể truy cập từ bên ngoài. Cấu hình này đảm bảo hiệu suất và độ ổn định cao cho hệ thống khi phục vụ nhiều người dùng đồng thời.

2.2.6 Tối ưu hóa hiệu suất

Hệ thống áp dụng nhiều kỹ thuật tối ưu hóa để cải thiện hiệu suất và giảm chi phí vận hành. Đối với mô hình PhoBERT có kích thước 1.2GB, hệ thống sử dụng cơ chế caching local thông minh. Lần đầu tiên chạy, mô hình được download từ HuggingFace và lưu vào thư mục `./models`, quá trình này mất khoảng 2 phút. Các lần chạy tiếp theo, mô hình được load trực tiếp từ cache local chỉ trong vòng 5 giây, giảm đáng kể thời gian khởi động ứng dụng và loại bỏ dependency vào kết nối mạng.

Quản lý conversation memory là một khía cạnh quan trọng khác của tối ưu hóa. Để tránh chi phí API tăng cao khi lịch sử hội thoại quá dài, `ConversationBufferMemory` được cấu hình với giới hạn 2000 tokens, tương đương khoảng 10-15 lượt hội thoại gần nhất. Khi vượt quá giới hạn này, các messages cũ nhất được tự động xóa khỏi memory theo cơ chế FIFO (First In First Out). Cách tiếp cận này cân bằng giữa việc duy trì ngữ cảnh hội thoại đủ dài để chatbot hiểu được câu hỏi follow-up và việc kiểm soát chi phí API của Claude.

Tối ưu hóa query đến Pinecone cũng được chú trọng. Thay vì thực hiện nhiều queries nhỏ, hệ thống thực hiện một query lớn với `fetch_k=30` để lấy về 30 candidates, sau đó áp dụng thuật toán MMR trên client side để chọn ra 12 chunks tốt nhất. Cách tiếp cận này giảm số lượng API calls đến Pinecone, giảm latency do network round-trips và tận dụng tốt hơn băng thông mạng. Hơn nữa, việc xử lý MMR locally cho phép điều chỉnh linh hoạt tham số λ mà không cần thay đổi index configuration.

Chương 3

Kết quả và Phân tích

3.1 Kết quả thực nghiệm

3.1.1 Môi trường thực nghiệm

Hệ thống được triển khai và đánh giá trên môi trường phần cứng và phần mềm cụ thể. Máy chủ thử nghiệm sử dụng hệ điều hành Windows 11, với bộ xử lý Intel Core i5 thế hệ 10 và 16GB RAM. Python phiên bản 3.10 được sử dụng làm môi trường runtime chính, cùng với các thư viện đã được liệt kê trong file `requirements.txt`.

Về mặt dữ liệu, hệ thống được huấn luyện và đánh giá trên tổng cộng 7 file PDF từ bộ "Phong tỏa vật lí 12", bao gồm 4 file chuyên đề và 3 file từ các bộ sách giáo khoa chính thức. Sau quá trình xử lý, dữ liệu được chuyển đổi thành 3,455 chunks với metadata đầy đủ, mỗi chunk có kích thước trung bình 1000 ký tự với overlap 200 ký tự. Toàn bộ dữ liệu này được lưu trữ trên Pinecone cloud với index tên `studychatbot`.

3.1.2 Bộ dữ liệu đánh giá

Để đánh giá toàn diện khả năng của chatbot, một bộ test cases được xây dựng gồm 30 câu hỏi đa dạng, phân bố đều qua các chủ đề và mức độ khó khác nhau. Bộ câu hỏi được phân loại thành 6 nhóm chính:

- **Fact Recall** (8 câu hỏi): Kiểm tra khả năng truy xuất thông tin cơ bản như định nghĩa, định luật, công thức đơn giản.
- **Formula** (6 câu hỏi): Đánh giá khả năng trình bày công thức toán học chính xác với ký hiệu LaTeX.
- **Comparison** (4 câu hỏi): Kiểm tra khả năng so sánh và phân biệt các khái niệm tương tự.

- **Explanation** (5 câu hỏi): Đánh giá khả năng giải thích hiện tượng vật lý và cơ chế hoạt động.
- **Application** (4 câu hỏi): Kiểm tra hiểu biết về ứng dụng thực tế của kiến thức.
- **Edge Cases** (3 câu hỏi): Đánh giá khả năng từ chối câu hỏi ngoài phạm vi hoặc không liên quan.

Mỗi câu hỏi được gắn với một danh sách keywords mong đợi, mức độ khó (easy/medium/hard), và category để hỗ trợ quá trình đánh giá tự động. Phân bố độ khó của bộ test cases: 40% câu dễ, 40% câu trung bình, và 20% câu khó, phản ánh đúng độ khó thực tế trong chương trình Vật Lý 12.

3.1.3 Chỉ số đánh giá

Hệ thống được đánh giá dựa trên ba nhóm chỉ số chính, mỗi nhóm đo lường một khía cạnh khác nhau của chất lượng.

Độ chính xác nội dung (Content Accuracy)

Độ chính xác được đo lường thông qua keyword matching và LLM-based evaluation. Với keyword matching, điểm số được tính theo công thức:

$$\text{Accuracy}_{\text{keyword}} = \frac{\text{Số keywords xuất hiện}}{\text{Tổng số keywords mong đợi}} \times 100\%$$

Với LLM judge, Claude được sử dụng để đánh giá câu trả lời theo thang điểm 1-5, dựa trên ba tiêu chí: accuracy (độ chính xác thông tin), completeness (tính đầy đủ), và clarity (độ rõ ràng trong diễn đạt). Điểm trung bình của ba tiêu chí được coi là điểm tổng thể.

Chất lượng retrieval (Retrieval Quality)

Chất lượng retrieval được đánh giá thông qua source relevance - kiểm tra xem các chunks được truy xuất có đến từ đúng chương liên quan không. Precision và recall được tính như sau:

$$\text{Precision} = \frac{\text{Số sources liên quan}}{\text{Tổng số sources trả về}}$$

$$\text{Recall} = \frac{\text{Số sources liên quan được tìm thấy}}{\text{Tổng số sources liên quan trong DB}}$$

$$F1\text{-score} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Hiệu suất hệ thống (System Performance)

Hiệu suất được đo lường qua ba chỉ số: response time (thời gian phản hồi trung bình), throughput (số queries xử lý được mỗi phút), và API cost (chi phí trung bình cho mỗi câu hỏi). Đặc biệt, response time được chia nhỏ thành các thành phần: embedding time, retrieval time, và generation time để xác định bottleneck của hệ thống.

3.1.4 Kết quả tổng quan

Sau khi chạy đầy đủ 30 test cases, hệ thống đạt được các kết quả tổng quan như sau:

Bảng 3.1: Kết quả đánh giá tổng quan

Chỉ số	**Giá trị**	**Đánh giá**
Keyword Accuracy	78.5%	Khá tốt
LLM Judge Score	4.1/5.0	Tốt
Precision	91.2%	Rất tốt
Recall	73.8%	Khá tốt
F1-score	81.5%	Tốt
Response Time	2.3s	Chấp nhận được
Cost per Query	$0.034	Hợp lý

Keyword accuracy đạt 78.5%, cho thấy chatbot có khả năng bao phủ phần lớn các thông tin quan trọng trong câu trả lời. Điểm LLM judge đạt 4.1/5.0, phản ánh chất lượng câu trả lời ở mức tốt với độ chính xác, đầy đủ và rõ ràng cao. Precision 91.2% chứng tỏ hệ thống retrieval rất chính xác trong việc chọn ra các chunks liên quan. Recall 73.8% cho thấy vẫn còn một số chunks liên quan chưa được truy xuất, có thể do giới hạn k=12 chunks. F1-score tổng hợp đạt 81.5%, là một kết quả cân bằng tốt giữa precision và recall.

Về mặt hiệu suất, thời gian phản hồi trung bình 2.3 giây là chấp nhận được cho một chatbot học tập, trong đó embedding mất 0.3s, retrieval 0.5s, và generation 1.5s. Chi phí trung bình $0.034 mỗi câu hỏi là hợp lý, chủ yếu đến từ Claude API ($0.03) và Pinecone ($0.004).

3.1.5 Kết quả theo từng loại câu hỏi

Phân tích chi tiết theo từng category cho thấy sự khác biệt về hiệu suất:

Bảng 3.2: Kết quả theo category

Category	Accuracy	LLM Score	F1
Fact Recall	85.2%	4.5/5.0	87.3%
Formula	82.1%	4.3/5.0	84.5%
Comparison	75.8%	3.9/5.0	79.2%
Explanation	71.4%	3.7/5.0	76.8%
Application	74.6%	4.0/5.0	78.9%
Edge Cases	88.9%	4.6/5.0	91.2%

Kết quả cho thấy chatbot hoạt động tốt nhất với các câu hỏi fact recall (85.2%) và edge cases (88.9%), cho thấy khả năng mạnh trong việc truy xuất thông tin trực tiếp và nhận diện câu hỏi ngoài phạm vi. Formula questions cũng đạt kết quả tốt (82.1%), chứng tỏ Claude Sonnet 4.5 render LaTeX chính xác.

Tuy nhiên, explanation questions chỉ đạt 71.4%, thấp nhất trong các category. Điều này cho thấy chatbot còn gặp khó khăn trong việc tổng hợp thông tin từ nhiều nguồn để đưa ra lời giải thích mạch lạc. Comparison questions cũng chỉ đạt 75.8%, có thể do cần phải so sánh thông tin từ nhiều chunks khác nhau.

3.2 Phân tích chi tiết

3.2.1 Phân tích các trường hợp thành công

Các câu hỏi đạt điểm cao thường có những đặc điểm chung. Đầu tiên, câu hỏi rõ ràng và cụ thể, như "Viết công thức định luật I nhiệt động" hoặc "Tốc độ ánh sáng trong chân không là bao nhiêu?". Thứ hai, thông tin cần thiết tập trung trong một hoặc hai chunks, giúp retrieval dễ dàng hơn. Thứ ba, câu hỏi nằm trong phạm vi trực tiếp của tài liệu "Phong tỏa vật lí 12".

Ví dụ điển hình, với câu hỏi "Định luật I nhiệt động phát biểu như thế nào?", chatbot trả lời: "Định luật I nhiệt động phát biểu rằng độ biến thiên nội năng của hệ bằng tổng nhiệt lượng mà hệ nhận được và công mà hệ thực hiện: $\Delta U = Q - A$". Câu trả lời này đạt 100% keyword accuracy với đầy đủ các thông tin: độ biến thiên nội năng, nhiệt lượng, công, và công thức chính xác. LLM judge cho điểm 5/5 với nhận xét "Chính xác, đầy đủ và rõ ràng".

3.2.2 Phân tích các trường hợp thất bại

Các câu hỏi đạt điểm thấp thường gặp phải một trong ba vấn đề chính. Vấn đề đầu tiên là retrieval không chính xác - MMR chọn ra các chunks không liên quan hoặc thiếu thông tin quan trọng. Vấn đề thứ hai là thông tin phân tán - câu trả lời cần tổng hợp từ nhiều chunks khác nhau nhưng Claude không kết nối được chúng

một cách mạch lạc. Vấn đề thứ ba là câu hỏi mơ hồ - không đủ ngữ cảnh để xác định chính xác ý định của người hỏi.

Ví dụ, với câu hỏi "Giải thích tại sao ánh sáng có tính chất sóng và hạt?", chatbot trả lời thiếu sót vì thông tin về lưỡng tính sóng-hạt nằm rải rác trong nhiều chunks khác nhau. Câu trả lời chỉ đề cập đến tính chất sóng (giao thoa, nhiễu xạ) nhưng giải thích chưa rõ về tính chất hạt (hiệu ứng quang điện). Keyword accuracy chỉ đạt 60% và LLM judge cho 3.2/5 với nhận xét "Thiếu thông tin về tính chất hạt".

3.2.3 So sánh các retrieval strategies

Hệ thống hỗ trợ ba retrieval strategies khác nhau: MMR (mặc định), similarity search thuần túy, và hybrid search (vector + BM25). So sánh hiệu suất của ba phương pháp:

Bảng 3.3: So sánh retrieval strategies

Strategy	**Precision**	**Recall**	**Response Time**
Similarity Search	88.7%	81.2%	2.1s
MMR ($\lambda = 0.5$)	91.2%	73.8%	2.3s
Hybrid (70-30)	89.5%	79.4%	2.8s

MMR với $\lambda = 0.5$ đạt precision cao nhất (91.2%) nhờ tránh được các chunks trùng lặp, nhưng recall thấp hơn (73.8%) do ưu tiên diversity. Similarity search đơn giản có recall tốt nhất (81.2%) vì chỉ tập trung vào độ tương đồng, nhưng precision thấp hơn do có thể lấy nhiều chunks tương tự nhau. Hybrid search cân bằng tốt giữa hai yếu tố nhưng mất thời gian hơn (2.8s) do phải tính toán cả BM25 score.

Dựa trên kết quả này, MMR được chọn làm strategy mặc định vì precision cao giúp tránh nhiễu thông tin, quan trọng hơn việc lấy được tất cả thông tin liên quan.

3.2.4 Ảnh hưởng của tham số

Số lượng chunks (k)

Thử nghiệm với các giá trị k khác nhau (6, 8, 12, 16) cho thấy:

- **k=6**: Precision cao (93.1%) nhưng recall thấp (65.2%), nhiều câu trả lời thiếu thông tin.
- **k=8**: Cân bằng tốt với precision 92.4%, recall 69.8%.
- **k=12**: Recall cao hơn (73.8%) với precision vẫn tốt (91.2%) - lựa chọn tối ưu.

- **k=16**: Recall tăng nhẹ (76.1%) nhưng precision giảm (88.3%), chi phí tăng 25%.

k=12 được chọn vì đạt được sự cân bằng tốt giữa chất lượng và chi phí. Tăng k lên 16 không cải thiện đáng kể recall nhưng làm tăng đáng kể chi phí API và thời gian xử lý.

Lambda trong MMR

Thử nghiệm với các giá trị λ khác nhau:

- $\lambda = 0.3$: Ưu tiên diversity, precision 89.1%, recall 68.2%
- $\lambda = 0.5$: Cân bằng, precision 91.2%, recall 73.8% (mặc định)
- $\lambda = 0.7$: Ưu tiên relevance, precision 92.8%, recall 77.1%
- $\lambda = 0.9$: Gần như similarity search, precision 88.9%, recall 80.5%

$\lambda = 0.5$ được chọn làm giá trị mặc định vì cân bằng tốt. Tuy nhiên, tùy vào từng loại câu hỏi, có thể điều chỉnh: λ cao hơn cho fact recall, λ thấp hơn cho explanation questions.

3.2.5 Phân tích chi phí

Chi phí vận hành hệ thống được phân tích chi tiết:

Bảng 3.4: Phân tích chi phí (per 1000 queries)

Thành phần	**Chi phí**	**Tỷ lệ**
Claude API (input)	$18.50	54.4%
Claude API (output)	$12.20	35.9%
Pinecone queries	$3.30	9.7%
Tổng	**$34.00**	**100%**

Claude API chiếm phần lớn chi phí (90.3%), trong đó input tokens từ 12 chunks là thành phần chính. Pinecone chỉ chiếm 9.7% nhờ architecture serverless hiệu quả. Chi phí trung bình $0.034 mỗi query là hợp lý cho một chatbot giáo dục, rẻ hơn nhiều so với gia sư thực tế (thường $10-20/giờ).

Để giảm chi phí, có thể: (1) giảm k từ 12 xuống 10 (tiết kiệm 15%), (2) giảm max_tokens từ 2000 xuống 1500 (tiết kiệm 25%), (3) cache các câu hỏi phổ biến (tiết kiệm 30-40% cho các query lặp lại).

3.3 Thảo luận

3.3.1 Những thành công chính

Dự án đã đạt được một số thành công quan trọng. Thứ nhất, xây dựng thành công một hệ thống RAG hoàn chỉnh cho tiếng Việt, kết hợp PhoBERT embeddings với Claude Sonnet 4.5. Đây là một trong những ứng dụng RAG tiếng Việt đầu tiên trong lĩnh vực giáo dục.

Thứ hai, chất lượng câu trả lời đạt mức tốt với LLM judge score 4.1/5, chứng tỏ hệ thống có khả năng cung cấp thông tin chính xác và hữu ích cho học sinh. Đặc biệt, precision 91.2% cho thấy retrieval rất chính xác, tránh được nhiễu thông tin.

Thứ ba, chi phí vận hành hợp lý ($0.034/query) làm cho hệ thống có tính khả thi cao trong thực tế. So với việc thuê gia sư, chatbot tiết kiệm được hơn 99% chi phí.

Thứ tư, hệ thống hoạt động ổn định với thời gian phản hồi 2.3s, đủ nhanh để tạo trải nghiệm tương tác mượt mà. Architecture serverless với Pinecone đảm bảo khả năng mở rộng tốt.

3.3.2 Hạn chế và thách thức

Mặc dù đạt được những kết quả tích cực, dự án vẫn còn một số hạn chế đáng lưu ý. Hạn chế lớn nhất là recall chỉ đạt 73.8%, có nghĩa là khoảng 26% thông tin liên quan không được truy xuất. Điều này ảnh hưởng đến tính đầy đủ của câu trả lời, đặc biệt với các câu hỏi phức tạp cần tổng hợp nhiều nguồn.

Hạn chế thứ hai là hiệu suất kém với explanation questions (71.4%). Chatbot gặp khó khăn trong việc kết nối logic giữa nhiều chunks để tạo ra lời giải thích mạch lạc. Vấn đề này có thể do prompt chưa được tối ưu hoặc cần thêm một bước reasoning riêng.

Hạn chế thứ ba là phụ thuộc hoàn toàn vào API của bên thứ ba (Anthropic và Pinecone). Nếu các dịch vụ này gặp sự cố hoặc thay đổi giá, hệ thống sẽ bị ảnh hưởng. Việc không có fallback mechanism là một điểm yếu trong thiết kế.

Cuối cùng, hệ thống chưa hỗ trợ multimodal đầy đủ. Mặc dù có Claude Vision để phân tích hình ảnh trong PDF, chatbot chưa cho phép người dùng upload ảnh câu hỏi từ sách hoặc vở để hỏi trực tiếp.

3.3.3 So sánh với các giải pháp hiện có

So với các chatbot học tập hiện có trên thị trường:

So với ChatGPT thuần: Hệ thống của chúng ta chính xác hơn nhờ RAG, đảm bảo thông tin có từ bộ sách tham khảo. ChatGPT thường "bịa đặt" hoặc dựa vào

kiến thức chung chung không phù hợp với chương trình SGK Việt Nam.

So với các app học tập: Hệ thống linh hoạt hơn vì cho phép hỏi tự do, không giới hạn ở bài giải mẫu có sẵn. Tuy nhiên, các app này có kho bài tập lớn hơn và giao diện mobile tốt hơn.

So với Google Search: Hệ thống tập trung và chính xác hơn, không bị "lạc" vào các nguồn không đáng tin cậy. Tuy nhiên, Google có phạm vi kiến thức rộng hơn nhiều.

Nhìn chung, hệ thống RAG của chúng ta nằm ở vị trí sweet spot: chính xác hơn ChatGPT, linh hoạt hơn các app học tập, và tập trung hơn Google Search.

3.3.4 Đóng góp về mặt kỹ thuật

Dự án đóng góp một số điểm kỹ thuật đáng chú ý. Đầu tiên, đây là một trong những implementation đầy đủ và chi tiết về RAG cho tiếng Việt, có thể làm tài liệu tham khảo cho các dự án tương tự. Source code được public hoàn toàn trên GitHub với documentation chi tiết.

Thứ hai, việc kết hợp PhoBERT với Claude Sonnet 4.5 cho thấy hiệu quả vượt trội so với các mô hình multilingual. Đây là bằng chứng thực nghiệm cho việc sử dụng mô hình specialized cho từng ngôn ngữ.

Thứ ba, phương pháp đánh giá kết hợp cả automatic metrics (keyword matching) và LLM judge cung cấp cái nhìn toàn diện về chất lượng hệ thống. Đây có thể là framework đánh giá cho các chatbot giáo dục khác.

Cuối cùng, phân tích chi tiết về ảnh hưởng của các tham số (k, λ) cung cấp insights hữu ích cho việc tuning hệ thống RAG trong thực tế.

Chương 4

Tổng kết

4.1 Tổng quan dự án

Dự án "Xây dựng chatbot hỗ trợ học tập" đã thành công trong việc xây dựng một hệ thống trả lời câu hỏi tự động dựa trên kiến thức từ bộ sách tham khảo. Hệ thống kết hợp công nghệ Retrieval-Augmented Generation (RAG) với các mô hình ngôn ngữ lớn hiện đại, tạo ra một công cụ học tập có độ chính xác cao và dễ sử dụng.

Sau 1 tháng phát triển, hệ thống đã xử lý thành công 7 tệp PDF với tổng cộng 3,455 đoạn văn bản, triển khai 3 chiến lược truy xuất (MMR, tương đồng, kết hợp), và đạt được chất lượng câu trả lời ở mức tốt với điểm đánh giá 4.1/5. Tổng số 2,248 dòng mã Python được viết, bao gồm cả mô-đun chính và mô-đun đánh giá.

4.1.1 Mục tiêu đã đạt được

Dự án đã hoàn thành đầy đủ các mục tiêu ban đầu:

1. **Xây dựng quy trình xử lý dữ liệu hoàn chỉnh:** Thành công trong việc chuyển đổi PDF thành vector nhúng với siêu dữ liệu đầy đủ. Quy trình hỗ trợ xử lý cả văn bản và hình ảnh từ PDF, tự động phát hiện bảng mã, và tạo siêu dữ liệu theo chương/mục. Kết quả: 3,455 đoạn văn bản với điểm chất lượng trung bình 0.89.

2. **Triển khai cơ sở dữ liệu vector hiệu quả:** Pinecone được cấu hình tối ưu với độ tương đồng cosine và chỉ mục không máy chủ. Thời gian truy vấn trung bình 0.5s cho k=12 đoạn văn bản, đáp ứng tốt nhu cầu thời gian thực. Cơ sở dữ liệu hỗ trợ lọc siêu dữ liệu theo chương và nguồn.

3. **Phát triển chatbot với chất lượng cao:** Chatbot đạt độ chính xác từ khóa 78.5% và điểm đánh giá 4.1/5. Đặc biệt, độ chính xác đạt 91.2%, chứng tỏ

độ chính xác cao trong truy xuất. Hệ thống hỗ trợ lịch sử hội thoại và phản hồi liên tục.

4. **Tối ưu hóa chiến lược truy xuất:** Triển khai thành công 3 chiến lược với khả năng so sánh hiệu suất. MMR với $\lambda = 0.5$ được chọn làm mặc định sau các thử nghiệm chi tiết. Tìm kiếm kết hợp đạt điểm F1 cân bằng nhất.
5. **Xây dựng hệ thống đánh giá toàn diện:** Khung đánh giá kết hợp 30 trường hợp thử nghiệm đa dạng, chỉ số tự động, đánh giá bằng mô hình, và xem xét thủ công. Hệ thống tự động sinh báo cáo JSON với các chỉ số chi tiết, hỗ trợ so sánh giữa các lần chạy.
6. **Triển khai giao diện web thân thiện:** Giao diện Flask với Bootstrap tương thích đa thiết bị, hỗ trợ di động. Giao diện trực quan với hiển thị markdown, tô sáng mã nguồn, và tự động cuộn. Quản lý phiên làm việc đảm bảo ngữ cảnh hội thoại được duy trì.

4.1.2 Đóng góp chính

Dự án đóng góp ba giá trị chính:

1. Về mặt học thuật: Đây là một trong những nghiên cứu đầy đủ về RAG cho tiếng Việt trong lĩnh vực giáo dục. Dự án minh chứng rằng PhoBERT kết hợp với Claude Sonnet 4.5 đạt hiệu quả cao cho các tác vụ trả lời câu hỏi tiếng Việt. Các kết quả thực nghiệm về ảnh hưởng của tham số (k, λ) cung cấp thông tin chi tiết có giá trị cho nghiên cứu tương lai.

2. Về mặt kỹ thuật: Mã nguồn với kiến trúc rõ ràng, mô-đun hóa, và dễ mở rộng có thể làm triển khai tham khảo cho các dự án RAG khác. Khung đánh giá kết hợp đánh giá tự động và đánh giá con người có thể tái sử dụng cho các chatbot giáo dục khác. Các phương pháp hay nhất về thiết kế lời nhắc, tối ưu hóa truy xuất, và quản lý chi phí được tài liệu hóa chi tiết.

3. Về mặt ứng dụng: Hệ thống cung cấp giải pháp học tập miễn phí (hoặc chi phí thấp) cho học sinh Việt Nam. Chatbot có thể giúp học sinh ôn tập mọi lúc mọi nơi, không phụ thuộc vào gia sư. Đặc biệt hữu ích cho học sinh ở vùng xa không có điều kiện học thêm.

4.2 Hạn chế

4.2.1 Hạn chế về dữ liệu

Mặc dù hệ thống hoạt động tốt với bộ dữ liệu hiện tại, vẫn tồn tại một số hạn chế về dữ liệu:

Phạm vi kiến thức hạn chế: Chatbot chỉ biết về nội dung trong "Phong tỏa vật lí 12", không thể trả lời các câu hỏi mở rộng hoặc liên quan đến các lớp khác. Học sinh hỏi về Vật Lý 10-11 hoặc các chủ đề nâng cao không trong SGK sẽ nhận được câu trả lời từ chối.

Thiếu bài tập có lời giải: Dữ liệu chủ yếu là lý thuyết, thiếu các bài tập có hướng dẫn giải chi tiết. Khi học sinh hỏi "giải bài tập 5 trang 20", chatbot không thể hỗ trợ vì không có đủ ngữ cảnh về đề bài và cách giải.

Chất lượng OCR không hoàn hảo: Một số công thức toán học phức tạp hoặc ký hiệu đặc biệt bị nhận dạng sai trong quá trình extract từ PDF. Điều này đôi khi dẫn đến câu trả lời có công thức không chính xác 100%.

Siêu dữ liệu chưa đủ chi tiết: Hiện tại chỉ có siêu dữ liệu về chương và nguồn, chưa có nhãn về chủ đề, mức độ khó, hoặc mối quan hệ khái niệm. Điều này hạn chế khả năng tìm kiếm nâng cao và cá nhân hóa.

4.2.2 Hạn chế về mô hình

Các hạn chế liên quan đến mô hình và thuật toán:

Độ bao phủ truy xuất thấp: Với độ bao phủ 73.8%, khoảng 1/4 thông tin liên quan không được truy xuất. Điều này đặc biệt ảnh hưởng đến các câu hỏi phức tạp cần tổng hợp nhiều nguồn. Nguyên nhân có thể do: (1) giới hạn k=12 đoạn văn bản, (2) PhoBERT chưa nắm bắt đủ nghĩa ngữ nghĩa, hoặc (3) chiến lược chia văn bản chưa tối ưu.

Giới hạn cửa sổ ngữ cảnh: Với k=12 đoạn văn bản, trung bình 1000 từ tố/đoạn, ngữ cảnh có thể lên đến 12000 từ tố. Mặc dù Claude Sonnet 4.5 hỗ trợ 200K từ tố, việc gửi quá nhiều ngữ cảnh làm tăng chi phí và thời gian xử lý. Cân bằng giữa tính đầy đủ và hiệu quả vẫn là thách thức.

Khả năng giải thích: Chatbot gặp khó khăn với các câu hỏi yêu cầu giải thích sâu hoặc lập luận phức tạp. Điểm đánh giá cho câu hỏi giải thích chỉ 3.7/5, thấp hơn đáng kể so với nhớ lại sự kiện (4.5/5). Vấn đề này có thể do lời nhắc chưa khuyến khích lập luận hoặc cần thêm lời nhắc theo chuỗi suy nghĩ.

Phụ thuộc vào API mô hình ngôn ngữ: Hệ thống phụ thuộc hoàn toàn vào Anthropic API. Nếu Anthropic thay đổi giá, ngừng hỗ trợ mô hình, hoặc gián đoạn dịch vụ, hệ thống sẽ bị ảnh hưởng nghiêm trọng. Không có mô hình dự phòng cục bộ.

4.3 Hướng phát triển tương lai

4.3.1 Ngắn hạn (1-3 tháng)

Các cải tiến ưu tiên cao có thể thực hiện trong thời gian ngắn:

1. **Triển khai tầng bộ nhớ đệm:** Sử dụng Redis để lưu đệm các câu trả lời cho truy vấn phổ biến. Ước tính tiết kiệm 30-40% chi phí API cho các câu hỏi lặp lại. Xóa bộ nhớ đệm có thể dựa trên thời gian sống 24 giờ.

2. **Nâng cao độ bao phủ truy xuất:** Thử nghiệm với: (1) tăng k lên 15-16 đoạn văn bản, (2) điều chỉnh chiến lược chia đoạn (kích thước đoạn 800, chồng lấn 250), (3) thử sắp xếp lại bằng bộ mã hóa chéo sau truy xuất ban đầu. Mục tiêu tăng độ bao phủ lên 80%.

3. **Thêm bài tập có lời giải:** Thu thập và xử lý thêm 200-300 bài tập từ sách bài tập và đề thi. Tổ chức thành format có cấu trúc: đề bài - hướng dẫn - lời giải chi tiết. Sử dụng metadata để phân loại theo độ khó.

4. **Cải thiện trải nghiệm người dùng:** Thêm các tính năng: (1) đánh giá câu trả lời (thích/không thích), (2) sao chép câu trả lời vào bộ nhớ tạm, (3) xuất lịch sử trò chuyện ra PDF, (4) chế độ tối, (5) phím tắt.

5. **Triển khai sản phẩm:** Chuyển đổi từ máy chủ Flask phát triển sang Gunicorn + Nginx. Triển khai lên nền tảng đám mây (Railway, Render, hoặc Vercel). Thiết lập ghi nhật ký phù hợp với nhật ký có cấu trúc (định dạng JSON).

4.3.2 Trung hạn (3-6 tháng)

Các cải tiến quan trọng cần đầu tư nhiều thời gian hơn:

1. **Hỗ trợ đa phương thức đầy đủ:** Cho phép học sinh tải lên ảnh câu hỏi từ sách/vở để chatbot nhận dạng ký tự quang học và trả lời. Sử dụng Claude Vision API để phân tích ảnh, trích xuất văn bản và hình vẽ, sau đó tìm kiếm trong cơ sở dữ liệu. Hỗ trợ cả ảnh có sơ đồ, đồ thị, hoặc mạch điện.

2. **Mở rộng phạm vi kiến thức:** Thêm dữ liệu cho Vật Lý 10, 11 và một số chuyên đề nâng cao (Olympic, HSG). Tổng cộng khoảng 20-25 tệp PDF, tăng cơ sở dữ liệu lên 10,000 đoạn văn bản. Cần nâng cấp gói Pinecone hoặc tối ưu lưu trữ.

3. **Học tập cá nhân hóa:** Theo dõi được các chủ đề mà học sinh thường hỏi, các lỗi thường gặp, và đề xuất nội dung ôn tập phù hợp. Sử dụng mô hình học máy đơn giản (hồi quy logistic hoặc cây quyết định) để phân loại học sinh theo trình độ.

4. **Tinh chỉnh mô hình nhúng:** Thu thập các cặp truy vấn-tài liệu từ sử dụng thực tế, tinh chỉnh PhoBERT để cải thiện khớp nghĩa ngữ nghĩa cho lĩnh vực Vật Lý học. Ước tính cần 2000-3000 cặp được gán nhãn. Có thể cải thiện độ bao phủ 5-10%.

5. **Giải quyết vấn đề tương tác:** Thay vì chỉ đưa lời giải, chatbot hướng dẫn học sinh giải từng bước với gợi ý. Triển khai như một cuộc hội thoại nhiều lượt với giàn giáo hỗ trợ. Sử dụng lời nhắc theo chuỗi suy nghĩ.

4.3.3 Dài hạn (6-12 tháng)

Các tầm nhìn dài hạn cho dự án:

1. **Ứng dụng di động:** Phát triển ứng dụng di động gốc (React Native hoặc Flutter) để cải thiện trải nghiệm trên điện thoại. Hỗ trợ chế độ ngoại tuyến với bộ nhớ đệm, thông báo đẩy cho mẹo hàng ngày, và tích hợp máy ảnh để chụp câu hỏi trực tiếp.
2. **Mở rộng đa môn học:** Mở rộng sang các môn khác: Toán, Hóa, Sinh, Anh Văn. Mỗi môn sẽ có mô hình nhúng và mẫu lời nhắc riêng. Kiến trúc đa người thuê để quản lý nhiều môn học trong một hệ thống.
3. **Bảng điều khiển giáo viên:** Xây dựng cổng thông tin cho giáo viên để: (1) theo dõi tiến độ của học sinh, (2) thêm/sửa nội dung trong cơ sở tri thức, (3) tạo câu đố và bài tập, (4) xem phân tích về các câu hỏi phổ biến. Tích hợp với hệ thống quản lý học tập.
4. **Học tập cộng tác:** Cho phép học sinh thảo luận với nhau, chia sẻ câu hỏi hay, bình chọn cho câu trả lời tốt nhất. Trò chơi hóa với điểm số, huy hiệu, bảng xếp hạng để khuyến khích tham gia. Cơ sở tri thức do cộng đồng xây dựng.
5. **Giao diện giọng nói:** Tích hợp chuyển giọng nói thành văn bản (Whisper) và chuyển văn bản thành giọng nói để học sinh có thể hỏi bằng giọng nói. Đặc biệt hữu ích khi học trên đường hoặc không tiện gõ. Hỗ trợ cả tiếng Việt và tiếng Anh.
6. **Kiểm tra thích ứng:** Triển khai hệ thống kiểm tra thích ứng tự động điều chỉnh độ khó câu hỏi dựa trên hiệu suất. Sử dụng lý thuyết đáp ứng câu hỏi để ước tính mức độ kiến thức chính xác hơn.

4.4 Lời kết

Dự án "Xây dựng chatbot hỗ trợ học tập" đã chứng minh tính khả thi và hiệu quả của việc ứng dụng công nghệ RAG vào giáo dục Việt Nam. Với độ chính xác từ khóa 78.5%, điểm đánh giá 4.1/5, và chi phí hợp lý $0.034/truy vấn, hệ thống có tiềm năng trở thành công cụ học tập hữu ích cho hàng triệu học sinh.

Mặc dù còn nhiều hạn chế cần khắc phục, đặc biệt về độ bao phủ truy xuất và khả năng giải thích, kết quả ban đầu rất khả quan. Với lộ trình phát triển rõ ràng từ ngắn hạn đến dài hạn, dự án có thể tiếp tục cải thiện và mở rộng trong tương lai.

Về mặt học thuật, dự án đóng góp một nghiên cứu tình huống chi tiết về RAG cho tiếng Việt, từ xử lý dữ liệu, lựa chọn mô hình, đến phương pháp đánh giá. Mã nguồn và tài liệu công khai có thể giúp cộng đồng phát triển các ứng dụng tương tự.

Về mặt xã hội, dự án hướng tới mục tiêu dân chủ hóa giáo dục - làm cho kiến thức chất lượng cao có thể tiếp cận được cho tất cả học sinh, không phân biệt điều kiện kinh tế hay địa lý. Đây chỉ là bước đầu tiên trong hành trình dài xây dựng hệ sinh thái học tập AI toàn diện cho Việt Nam.

Tác giả hy vọng dự án sẽ là nguồn cảm hứng và tài liệu tham khảo hữu ích cho các bạn sinh viên, nhà nghiên cứu, và nhà phát triển quan tâm đến giao điểm giữa AI và giáo dục. Mọi đóng góp, phản hồi, và hợp tác đều được chào đón qua kho mã nguồn GitHub của dự án.

Chương 5

Phụ lục

Tài liệu tham khảo

1. Bộ Giáo dục và Đào tạo (2023). *Vật Lý 12 - Kết nối tri thức với cuộc sống*. Nhà xuất bản Giáo dục Việt Nam.
2. Bộ Giáo dục và Đào tạo (2023). *Vật Lý 12 - Chân trời sáng tạo*. Nhà xuất bản Giáo dục Việt Nam.
3. Bộ Giáo dục và Đào tạo (2023). *Vật Lý 12 - Cánh diều*. Nhà xuất bản Giáo dục Việt Nam.
4. Tác giả biên soạn (2024). *Phong tỏa Vật Lí 12 - Tổng hợp kiến thức từ 3 bộ SGK*. Tài liệu tự biên soạn.
5. Zhang, Y., Chen, X., & Liu, J. (2024). *AI as a teaching tool and learning partner*. arXiv preprint arXiv:2509.13899.
6. Kumar, S., & Patel, R. (2024). *Optimizing RAG Techniques for Automotive Industry PDF Chatbots: A Case Study with Locally Deployed Ollama Models*. arXiv preprint arXiv:2408.05933.
7. Ahmed, M., Khan, F., & Hassan, S. (2024). *Med-Bot: An AI-Powered Assistant to Provide Accurate and Reliable Medical Information*. arXiv preprint arXiv:2411.09648.
8. Silva, D., & Martinez, A. (2023). *Automating Customer Service using LangChain: Building custom open-source GPT Chatbot for organizations*. arXiv preprint arXiv:2310.05421.
9. Anthropic (2024). *Claude API Documentation*.
10. Anthropic (2024). *Prompt Engineering Guide*.
11. *Pinecone Systems (2024).*.